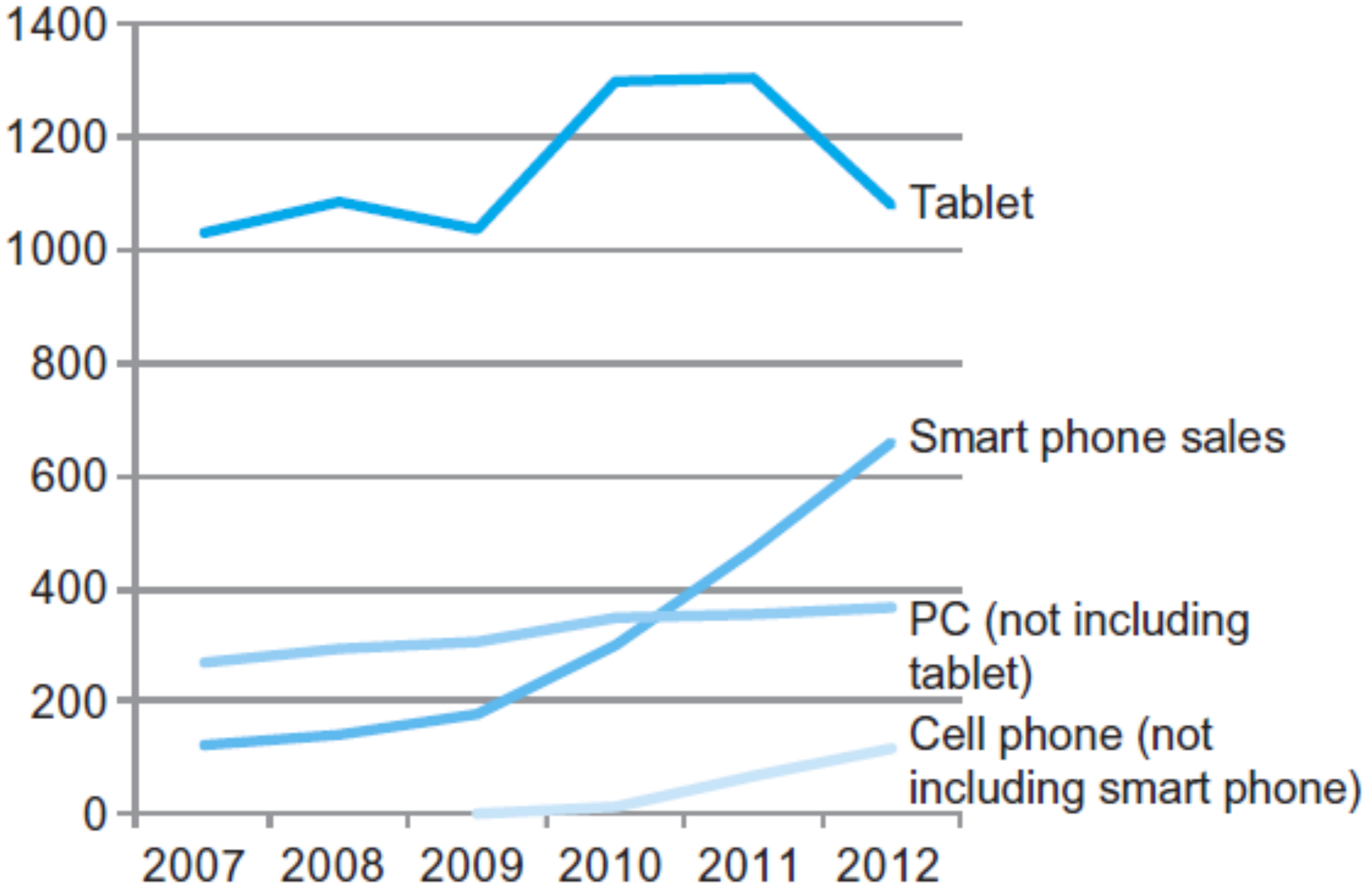
Capitolo 1

Classi di Computer

- **Personal Computer**
  - Scopi generali, molti software diversi
  - Soggetto a tradeoff costi/prestazioni
- **Server**
  - Basati su collegamenti di rete
  - Elevata capacità, prestazioni, affidabilità
  - Da piccoli server a interi capannoni
- **Supercomputer**
  - Complesse elaborazioni scientifiche/ingegneristiche
  - Massima capacità, ma rappresentano una piccola frazione del mercato dei computer
- **Computer Dedicati (*Embedded*)**
  - Nascosti come componenti di sistemi complessi
  - Rigorosi vincoli su potenza/prestazioni/costi

Era PostPC


1400
1200
1000
800
600
400
200
0
2007
2008
2009
2010
2011
2012
Tablet
Smart phone sales
PC (not including tablet)
Cell phone (not including smart phone)

Era PostPC

- **Dispositivi Mobile Personali**
  - Alimentati a batteria
  - Connessi a Internet
  - Poche centinaia di euro
  - Smart Phone, Tablet, Smart Watch
- **Cloud Computing**
  - Computer grandi quanto magazzini (*warehouse scale computers, WSC*)
  - Software come servizio (*Software as a Service, SaaS*)
  - Parte del software è in esecuzione su un dispositivo mobile personale e parte sul Cloud
  - Amazon, Google, Microsoft

Comprendere le prestazioni

- **Algoritmo**
  - Determina il numero di operazioni eseguite
- **Linguaggio di programmazione, compilatore, architettura**
  - Determina il numero di istruzioni macchina eseguite per operazione
- **Processore e memoria**
  - Determinano quanto velocemente le istruzioni sono eseguire
- **Sistema di I/O (incluso il SO)**
  - Determina quanto velocemente le operazioni di I/O sono eseguite

Legge di Moore

Il numero di transistor su un chip raddoppierà ogni anno

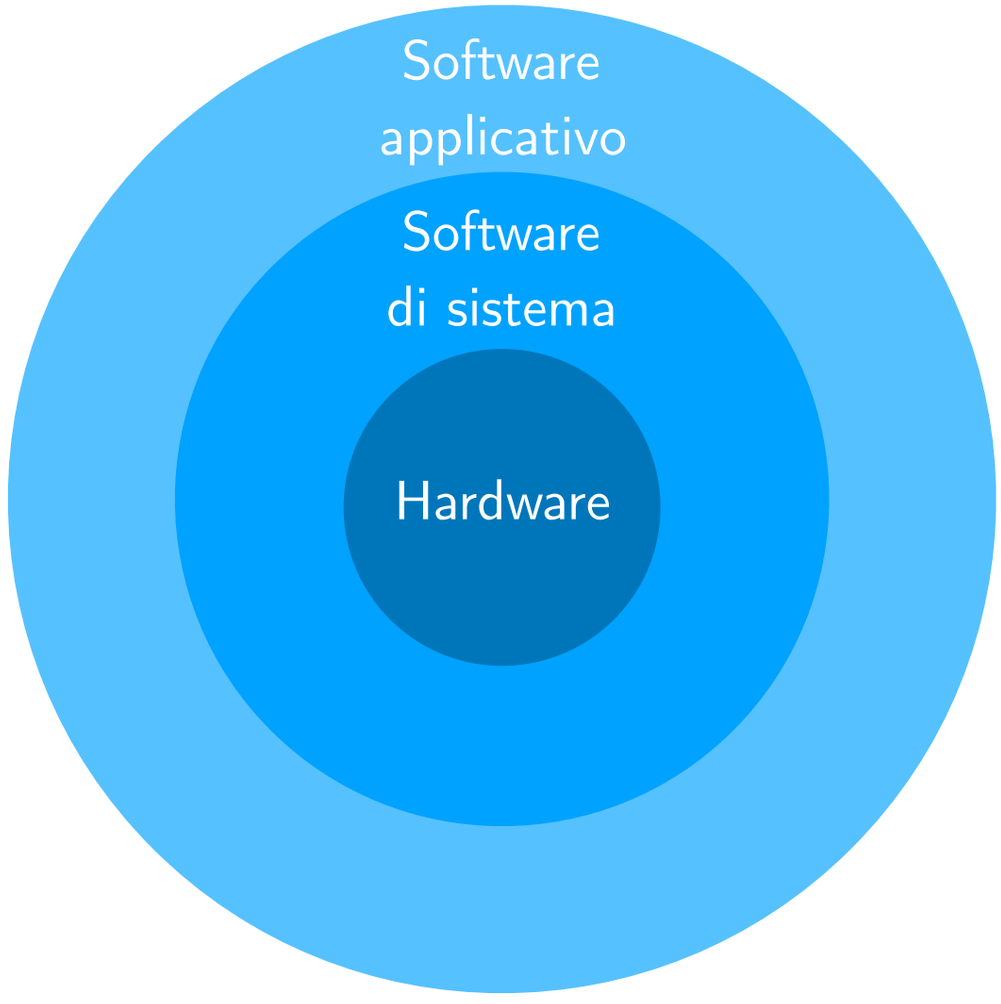
- Il numero di transistor su un chip raddoppia ogni 18 mesi
- Il costo di un chip è rimasto pressoché invariato
- Maggior densità di transistor su chip significa linee elettriche più corte, con maggiori prestazioni
- La minor dimensione aumenta la flessibilità
- Ridotti requisiti di potenza di alimentazione e di raffreddamento
- Un minor numero di interconnessioni aumenta l'affidabilità
- Gordon Moore è stati il co-fondatore della Intel

Otto Grandi Idee

- Progettare per la **Legge di Moore**
- Usare **astrazioni** per semplificare il progetto
- Rendere il **caso comune veloce**
- Prestazioni tramite **parallelismo**
- Prestazioni tramite **pipelining**
- Prestazioni tramite **predizione**
- **Gerarchia** di memoria
- **Affidabilità** tramite ridondanza

Dietro al vostro programma

- **Software applicativo**
  - Scritto in un linguaggio ad alto livello (HLL)
- **Software di sistema**
  - **Compilatore**: traduce il codice HLL in codice macchina
  - **Sistema Operativo**: codice di servizio
    - Gestisce input/output
    - Gestisce la memoria e l'archiviazione
    - Schedula i task e gestisce la condivisione delle risorse
- **Hardware**
  - Processore, memoria, controllori I/O

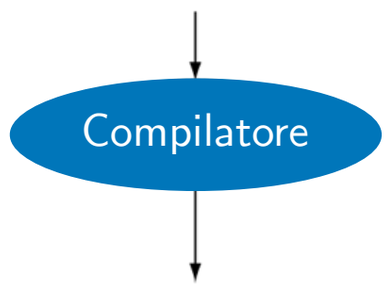


Livelli del codice di un programma

- **Linguaggio ad alto livello**
  - Livello di astrazione più vicino al dominio del problema
  - Fornisce produttività e portabilità
- **Linguaggio Assembly**
  - Rappresentazione testuale delle istruzioni
- **Rappresentazione Hardware**
  - Simboli binari (bit)
  - Istruzioni e dati codificati

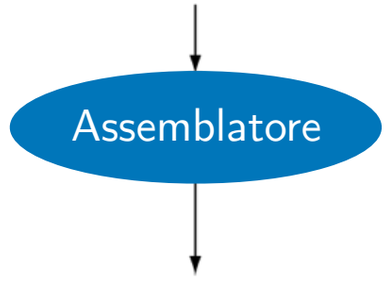
Programma in linguaggio ad alto livello (in C/C++)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```



Programma in linguaggio Assembly (per MIPS)

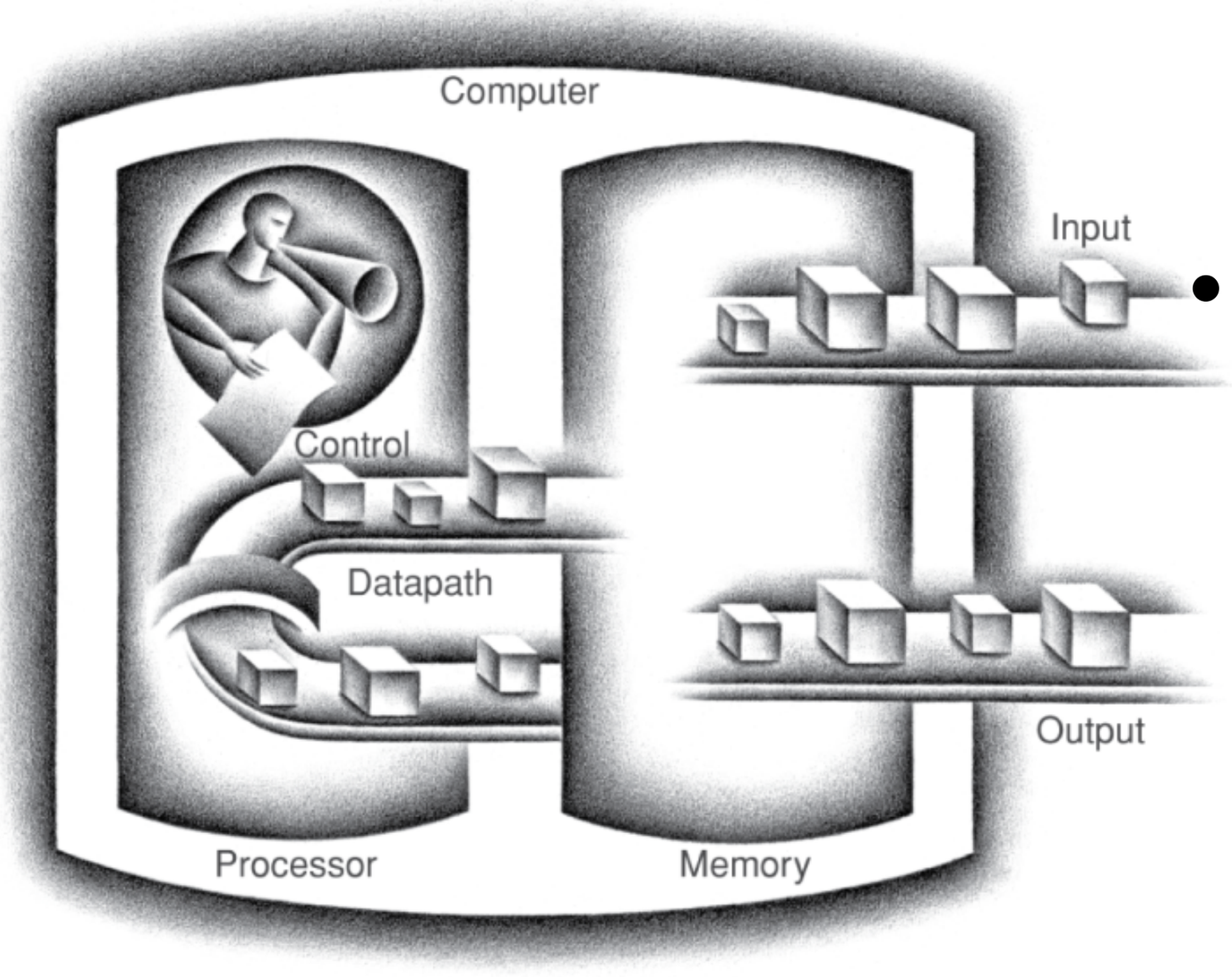
```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```



Programma in linguaggio macchina binario (per MIPS)

```
00000000101000010000000000011000
00000000000011000000110000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Componenti di un Computer

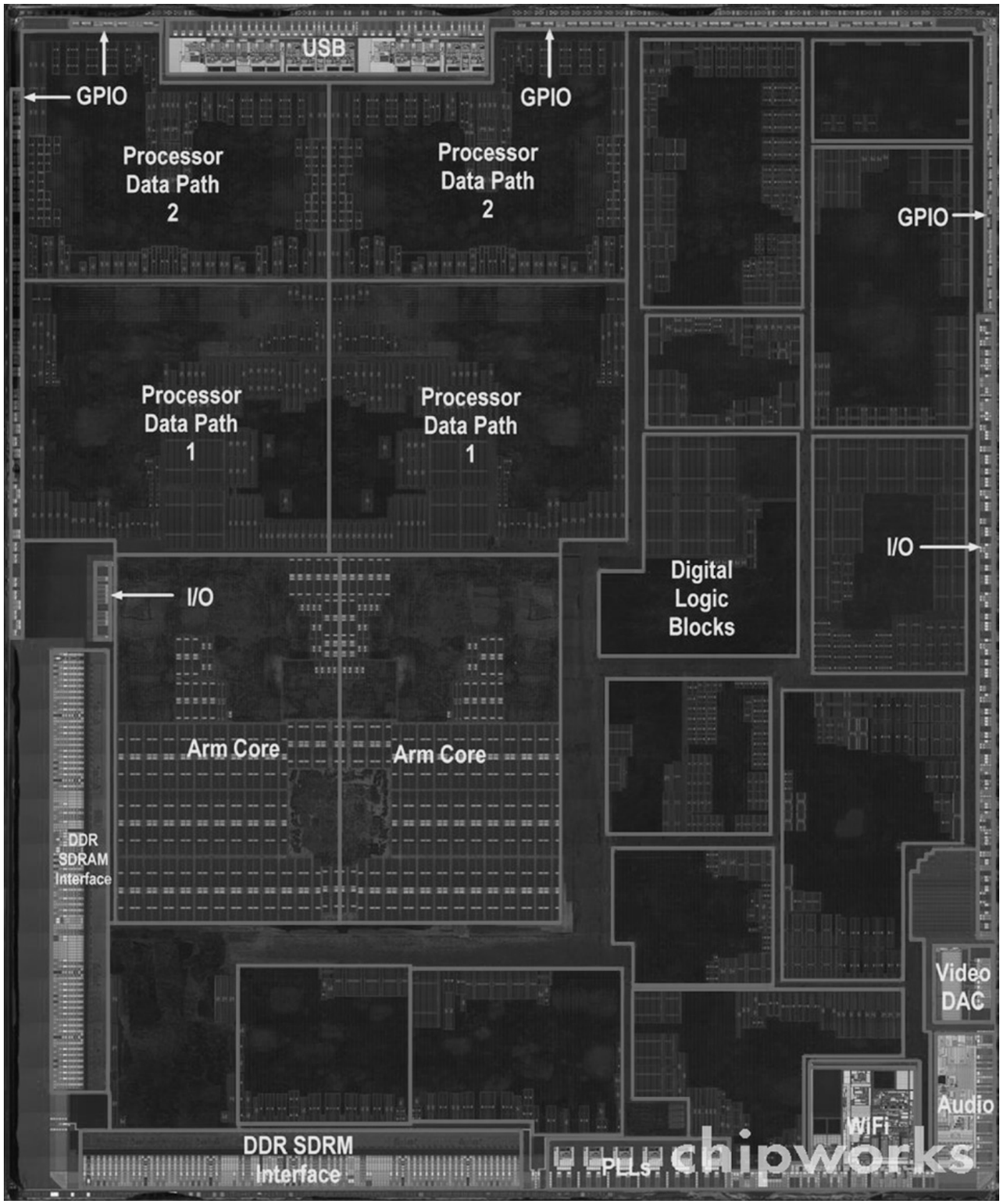


- Stessi componenti per tutti i tipi di computer
  - Desktop, server, embedded
- Input/output include
  - Dispositivi di interfaccia utente
    - Monitor, tastiera, mouse
- Dispositivi di archiviazione
  - Hard disk, CD/DVD, flash drive
- Adattatori di rete
  - Per comunicare con altri computer

Dentro la scatola

Dentro il processore (CPU)

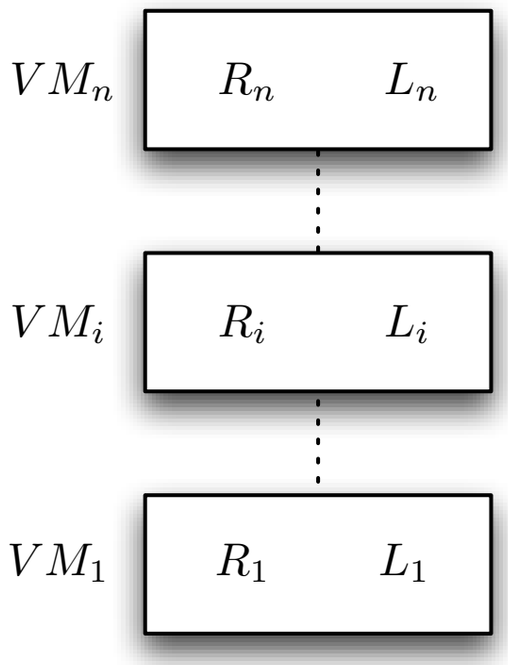
Apple A5

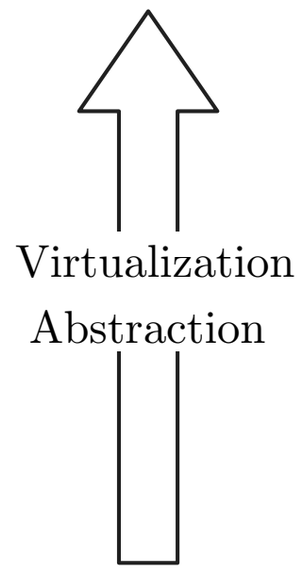


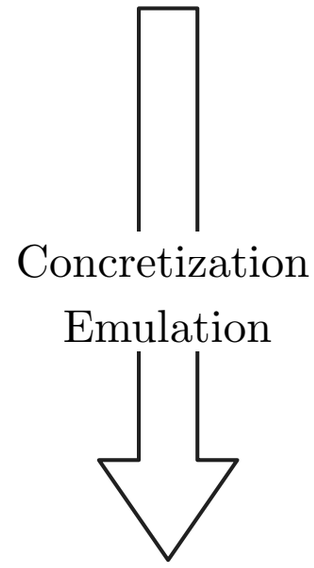
Livelli di astrazione

Astrazione:

- Utilizzata per gestire la complessità
- tipicamente distinta in livelli (*VMi*)
- ogni livello possiede il suo linguaggio L_i e le sue strutture dati R_i
- i livelli inferiori sono implementati in hardware
- i livelli superiori sono implementati in software







Livelli tipici:

5. Applicazioni
4. Assembler
3. Linguaggio Macchina (ISA)

Software

Hardware

2. Microarchitettura (firmware)
1. Logica digitale

Livello di logica digitale

- Linee e porte logiche (*gate*) realizzati tramite transistor
- Le porte logiche elaborano segnali binari (0/1)
- Le linee trasportano segnali binari (0/1)
- Tramite questi segnali è possibile rappresentare qualsiasi tipo di informazioni
- Interconnettendo porte logiche e linee è possibile realizzare funzioni complesse (es. moltiplicazioni di interi)
- Possibile realizzare elementi di memoria

Livello di microarchitettura

- Costruito sopra il livello della logica digitale, si occupa di interpretare ed eseguire le istruzioni del livello ISA

Livello del linguaggio macchina

- È il livello di macchina che appare al programmatore di sistema
- Comprende un insieme di *istruzioni* che di solito sono *diverse per ogni processore*
- La sintassi è adatta ad essere *interpretata facilmente* dal livello sottostante
- Le istruzioni sono *stringhe di bit* con *formato ben determinato* per permettere la facile individuazione di codici e operandi delle istruzioni
- I codici operativi (*opcode*) individuano l'operazione elementare che l'istruzione dovrà eseguire
- gli *operandi* si riferiscono alle locazioni che contengono i dati su cui eseguire l'operazione, e le locazioni dove memorizzare i risultati

Livello assembler

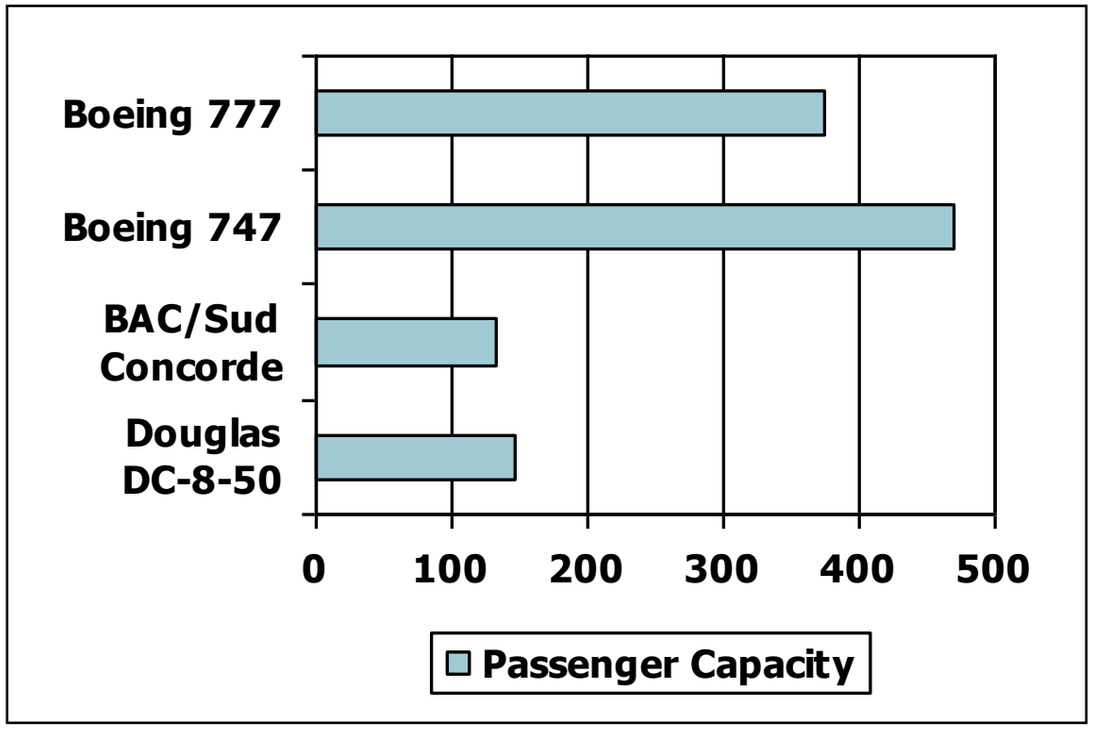
- Offre al programmatore di sistema una vista più "umana" del livello macchina:
  - istruzioni espresse con *stringhe di caratteri mnemoniche* invece di stringhe binarie
  - traduzione realizzata dall'*assemblatore* (uno speciale compilatore) rispetto al livello del linguaggio macchina
- L'*assemblatore* è stato uno dei primi software di sistema realizzato per facilitare la programmazione dei calcolatori

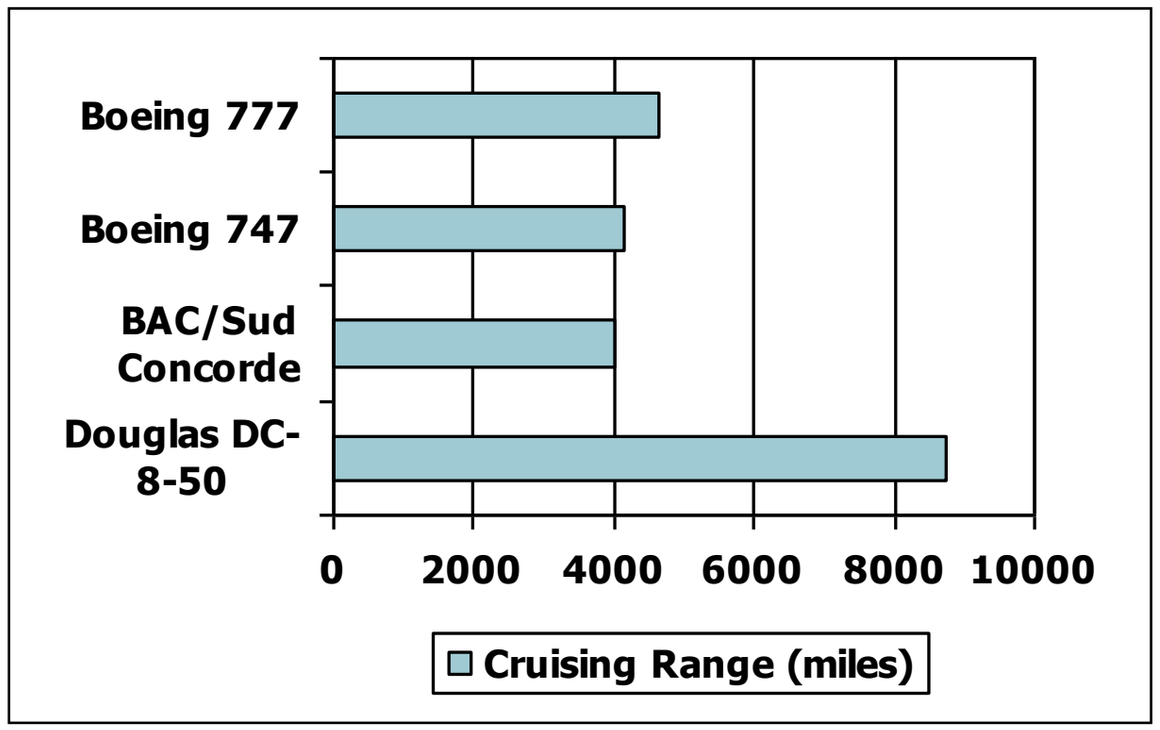
Livello delle applicazioni

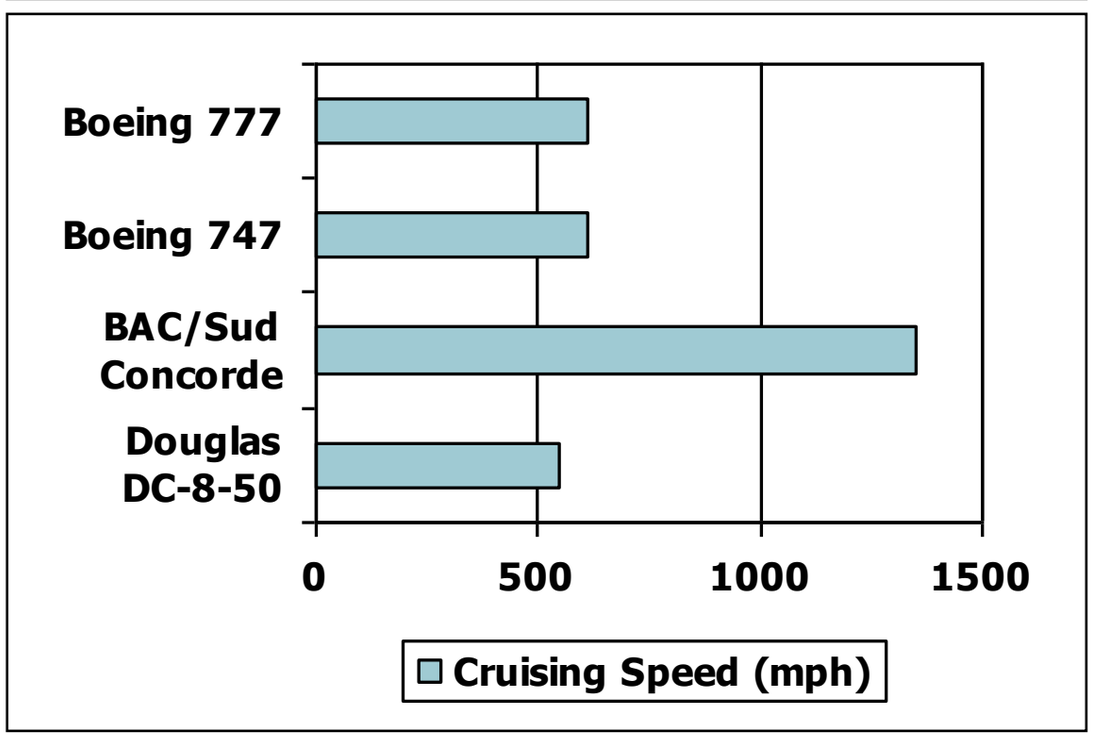
- Linguaggi ad alto livello: C, C++, Java, Python
- Permette al programmatore una maggiore astrazione rispetto ai livelli sottostanti della macchina
- Ha di solito bisogno di un traduttore (compilatore)
- E' il livello solitamente usato per produrre software
- Tale livello permette di realizzare la portabilità tra processori con diversa ISA
  - Basta che esista il compilatore/interprete implementato per la nuova ISA

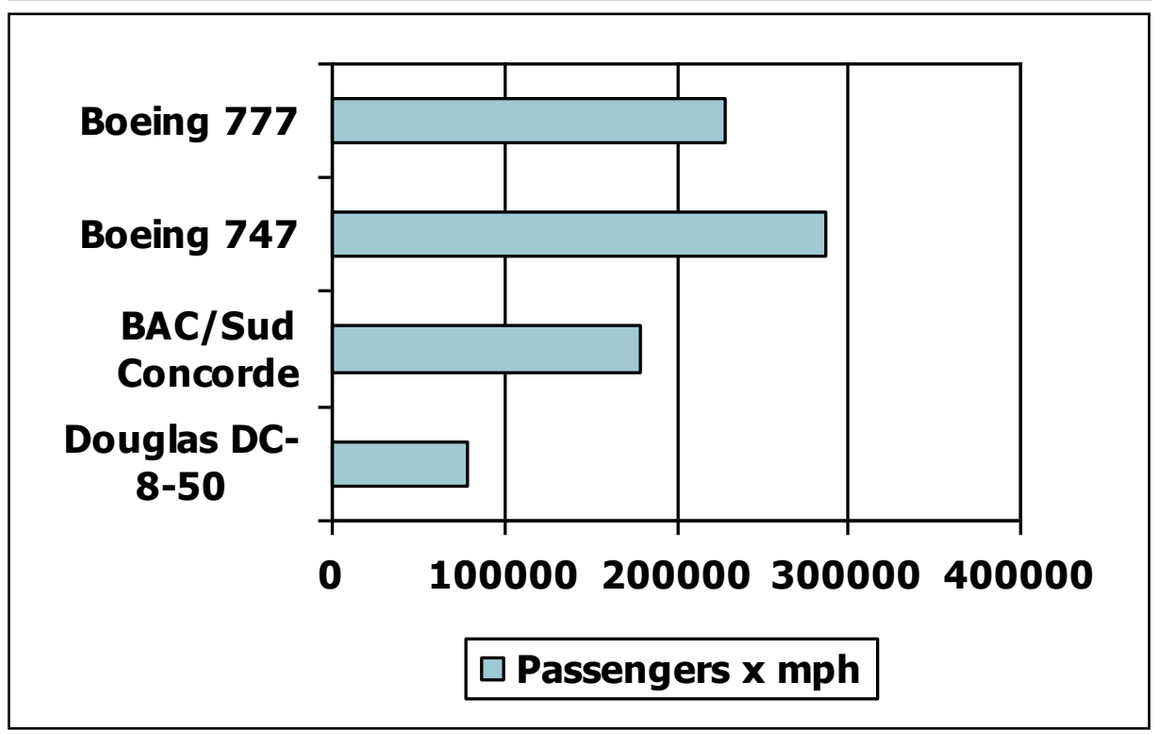
Definire le prestazioni

Quale aeroplano ha le migliori prestazioni?









Tempo di risposta e Throughput

- Tempo di risposta
  - Quanto si impiega per eseguire un task
- Throughput (banda)
  - Lavoro totale svolto per unità di tempo
    - Per esempio, task/transazioni/... all'ora
- Come cambiano il tempo di risposta e il throughput quando...
  - Si sostituisce il processore con una versione più veloce?
  - Si aggiungono più processori?
- Ci concentreremo sul tempo di risposta...

Performance relativa

- Si definisce Performance = 1/Tempo di esecuzione
- "X è *n* volte più veloce di Y"

$$\text{Performance}_X/\text{Performance}_Y = \text{Tempo di esecuzione}_Y/\text{Tempo di esecuzione}_X = n$$

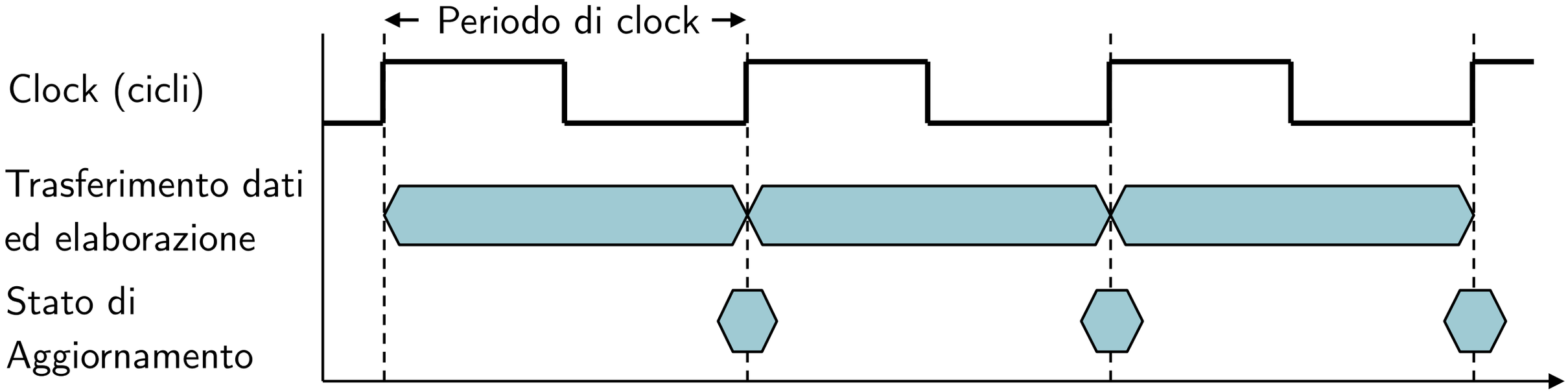
- Esempio: tempo impiegato a eseguire un programma
  - 10 secondi su A, 15 secondi su B
- Tempo di esecuzioneB / Tempo di esecuzioneA = 15/10 = 1.5
- Quindi A è 1.5 più veloce di B

Misurare il tempo di esecuzione

- Tempo trascorso
  - Tempo di risposta totale, tutto incluso
    - Elaborazione, I/O, overhead del SO, tempo di attesa
- Tempo di CPU
  - Tempo speso nell'esecuzione di un job
  - Comprende tempo di CPU utente e tempo di CPU di sistema
  - Programmi diversi subiscono diversamente le prestazioni di sistema e di CPU

Clock della CPU

- Le operazioni dell'hardware digitale sono governate da un clock a frequenza costante



- Periodo di clock: durata di un ciclo di clock
  - Per esempio, 250 ps = 0.25 ns = 250 x 10^{-12} s
- Frequenza di clock: cicli al secondo
  - Per esempio, 4.0 GHz = 4000 MHz = 4.0 x 10^{9} Hz

Tempo di CPU

Tempo di CPU = Cicli di clock della CPU x Periodo di clock =

$$= \frac{\text{Cicli di clock della CPU}}{\text{Frequenza di clock}}$$

- Prestazioni migliorate tramite:
  - Riduzione del numero di cicli di clock
  - Aumento della frequenza di clock
  - Il progettista hardware deve spesso bilanciare frequenza di clock e numero di cicli di clock

Esempio Tempo di CPU

- Computer A: 2 GHz, 10 s tempo di CPU
- Progettare il computer B:
  - riducendo il tempo di CPU a 6 s
  - si può avere un clock più veloce, ma incorrendo in 1.2 x cicli di clock
- Quanto veloce deve essere il clock del computer B?

$$\text{Frequenza di clock}_B = \frac{\text{Cicli di clock}_B}{\text{Tempo di CPU}_B} = \frac{1.2 \text{ x Cicli di clock}_A}{6\text{s}}$$

$$\text{Cicli di clock}_A = \text{Tempo di CPU}_A \text{ x Frequenza di clock}_A$$
$$= 10 \text{ s x } 2 \text{ GHZ} = 20 \text{ x } 10^9$$

$$\text{Frequenza di clock}_B = \frac{1.2 \text{ x } 20 \text{ x } 10^9}{6 \text{ s}} = \frac{24 \text{ x } 10^9}{6\text{s}} = 4 \text{ GHz}$$

Instruction Count e CPI

$$\text{Cicli di clock} = \text{Instruction Count x Cicli Per Istruzione (CPI)}$$

$$\text{Tempo di CPU} = \text{Instruction Count x CPI x Periodo di clock}$$

$$= \frac{\text{Instruction Count x CPI}}{\text{Frequenza di clock}}$$

- Instruction Count di un programma:
  - Determinato da programma, ISA e compilatore
- Cicli medi per istruzione
  - Determinati dall'hardware della CPU
  - Se istruzioni differenti hanno CPI differenti
    - Il CPI medio dipende dal mix delle istruzioni

Esempio CPI

- Computer A: periodo di clock = 250 ps, CPI = 2.0
- Computer B: periodo di clock = 500 ps, CPI = 1.2
- Stessa ISA
- Qual è il più veloce, e di quanto?

$$\text{Tempo di CPU}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Periodo di clock}_A$$
$$= I \times 2.0 \times 250 \text{ ps} = I \times 500 \text{ ps}$$
$$\text{Tempo di CPU}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Periodo di clock}_B$$
$$= I \times 1.2 \times 500 \text{ ps} = I \times 600 \text{ ps}$$
$$\frac{\text{Tempo di CPU}_B}{\text{Tempo di CPU}_A} = \frac{I \times 600 \text{ ps}}{I \times 500 \text{ ps}} = 1.2$$

CPI nei dettagli

- Se classi di istruzioni differenti impiegano un numero differente di cicli di clock

$$\text{Cicli di clock} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

- CPI medio pesato

$$\text{CPI} = \frac{\text{Cicli di clock}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \frac{\text{Instruction Count}_i}{\text{Instruction Count}} \right)$$

Esempio CPI

- Sequenze di codice compilato alternative usano le istruzioni nelle classi A, B e C

Classe	A	B	C
CPI per classe	1	2	3
IC sequenza 1	2	1	2
IC sequenza 2	4	1	1

- Sequenza 1: IC = 5
  - Cicli di clock

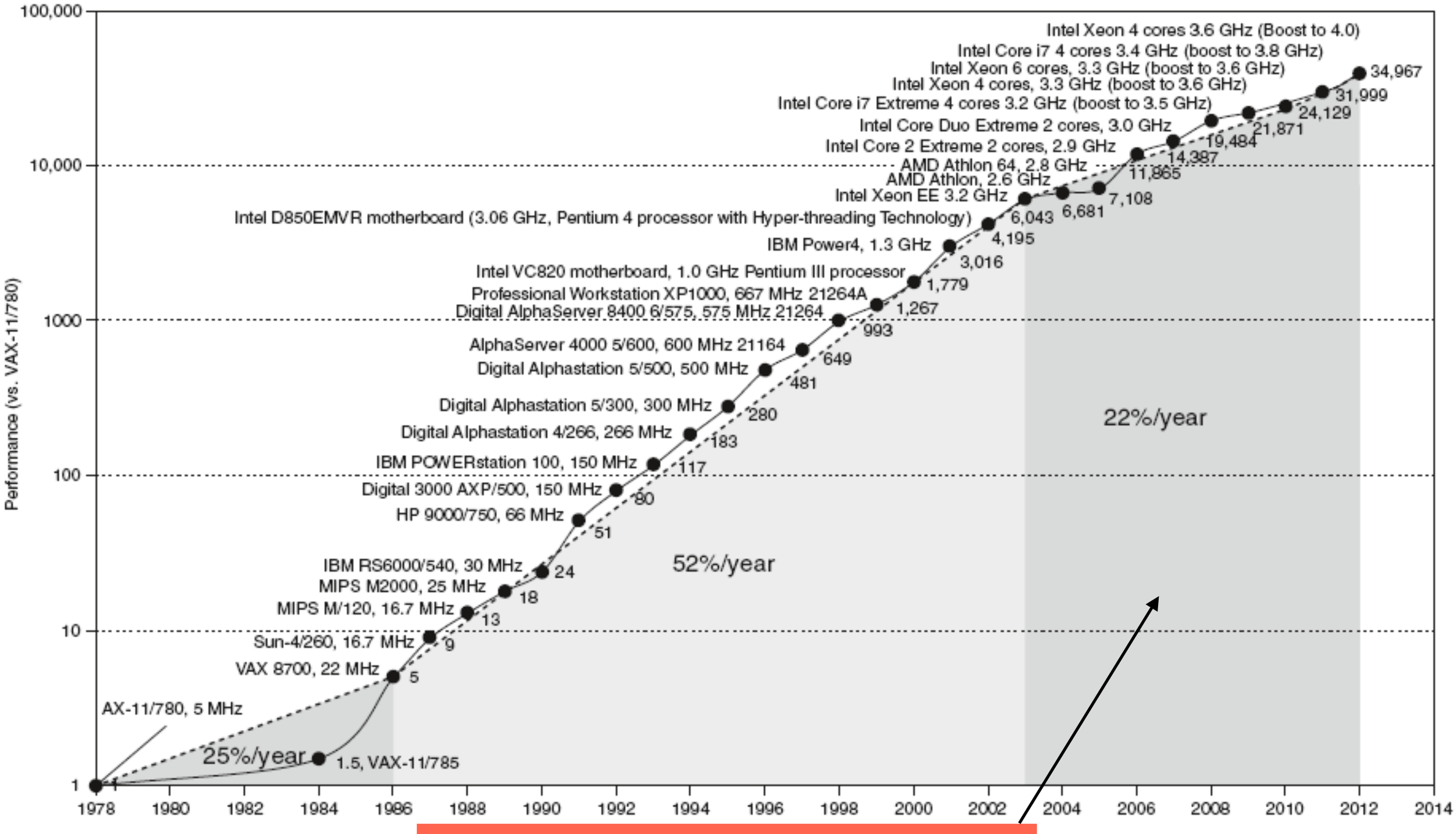
    = 2 x 1 + 1 x 2 + 2 x 3

    = 10
  - CPI medio = 10/5 = 2.0
- Sequenza 2: IC = 6
  - Cicli di clock

    = 4 x 1 + 1 x 2 + 1 x 3

    = 9
  - CPI medio = 9/6 = 1.5

Prestazioni uniprocessori



Vincolati da potenza, parallelismo a livello di istruzioni, latenza della memoria

Multiprocessori

- Microprocessori multicore
  - Più di un processore per chip
- Richiedono programmazione parallela esplicita
  - A differenza del parallelismo a livello di istruzioni
    - L'hardware esegue molteplici istruzioni alla volta
    - Nascosto al programmatore
- Difficile
  - programmare per le prestazioni
  - bilanciare il carico
  - ottimizzare le comunicazioni e la sincronizzazione

Sommario sulle prestazioni

$$\text{Tempo CPU} = \frac{\text{Istruzioni}}{\text{Programma}} \times \frac{\text{Cicli di clock}}{\text{Istruzione}} \times \frac{\text{Secondi}}{\text{Ciclo di clock}}$$

- Le prestazioni dipendono da:
  - Algoritmo: determina IC, possibilmente CPI
  - Linguaggio di programmazione: determina IC, CPI
  - Compilatore: determina IC, CPI
  - ISA: determina IC, CPI e periodo di clock